IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR U.S. LETTERS PATENT

Title:

SYSTEM AND METHOD FOR USER ADAPTIVE SOFTWARE INTERFACE

Inventor:

Michael A. Tremblay
3181 Blue Mountain Court
Loveland, Colorado 80537
Citizenship: US

SYSTEM AND METHOD FOR USER ADAPTIVE SOFTWARE INTERFACE

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] The present application is a continuation of U.S. Patent Serial No. 09/911,337, filed July 23, 2001, entitled "SYSTEM AND METHOD FOR USER ADAPTIVE SOFTWARE INTERFACE," which is incorporated herein by reference.

TECHNICAL FIELD

[0002] The present invention is related to a system and method for providing a software user interface and more particularly to a system and method for providing an adaptive software user interface.

BACKGROUND

[0003] Computer systems and consumer software applications have increasingly become more powerful and more complex. Consumer software applications frequently provide diverse functionality requiring complex interactions between a user and the user interface. The complexity of these interactions is quite often too cumbersome for most users. For example, certain procedures or routines associated with a consumer software application may be used relatively infrequently. A user may be required to relearn the various menu selections necessary to perform such procedures or routines due to the length of time involved between repetition of the procedures or routines.

[0004] To address these issues, software producers have developed various help topic lists. In general, an alphabetical list of help topics is presented to a user when the user clicks on an appropriate menu or icon. The user may scroll through the list. Alternatively, the user may perform a rudimentary text search to identify a specific topic in the list. However, this approach is problematic in many respects. In particular, it requires the software producer to identify each and every potential help topic before release of the software product. The help topic list is then sufficiently large that it does not allow user to quickly locate help topics of interest. Instead, the user must cull through numerous selections before being able to obtain desired information.

SUMMARY OF THE INVENTION

[0005] In one embodiment, the present invention is directed to a method for providing an adaptive computer user interface. The method may comprise the steps of: monitoring operating system events associated with input from a user to generate a series of operating system events; determining whether said series of operating system events is unrelated; and when said series of operating system events is unrelated, offering help to said user.

[0006] In another embodiment, the present invention is directed to a system for providing computer user interface. The system may comprise a means for monitoring user input; a means for determining whether a series of events is a unrelated series; and a means for offering assistance to said user, wherein said means for offering assistance is operable to offer assistance when said means for determining determines that said series of events is a unrelated series.

BRIEF DESCRIPTION OF THE DRAWINGS

[0007] FIGURE 1 depicts exemplary steps according to a preferred embodiment of the present invention.

[0008] FIGURE 2 depicts a block diagram of a computer system which is adapted to use the present invention.

DETAILED DESCRIPTION

[0009] FIGURE 1 depicts exemplary steps to illustrate an embodiment of the present invention. In this preferred embodiment of the present invention, various steps occur as operations or instructions performed by a modification to an operating system. In particular, operating system events are monitored to detect user actions. By doing so, this embodiment is capable of analyzing input associated with a plurality of applications, programs, and system components.

[0010] In step 101, a detector algorithm monitors user input information from the operating system until an input event occurs (a move event associated with a mouse, a keystroke, a peripheral click, and/or the like). In step 102, the detector algorithm adds the event to an event queue. In step 103, the input event is examined to determine whether it is an explicit request for help (e.g., pushing the F1 button or clicking on a help icon). If the input event is an explicit

help (e.g., pushing the F1 button or clicking on a help icon). If the input event is an explicit request for help, the detector algorithm proceeds to step 111. If the input event is not an explicit request for help, the detector algorithm determines whether this is the only event in the event queue (step 104). If this is the only event in the event queue, the detector algorithm returns to step 101.

[0011] If the event is not the only event in the event queue, the detector algorithm determines whether the series of events in the event queue is a pattern of unrelated events (step 105). This analysis attempts to determine whether the user is searching for information or may require help. The analysis may attempt to ascertain whether the user has accessed ordinary search facilities. Additionally or alternatively, this analysis attempts to determine whether the user has repeatedly accessed program functionality without completing any specific tasks. For example, the analysis may determine that a series of events is unrelated if a user accesses several menus without making a selection or accesses different programs. In this step, it may also be desirable to verify the timing relationship between events.

[0012] If events are separated by too much time, analysis of the events of the series may not provided useful information and should be cleared from the queue (step 106). If the set of events in the event queue is determined to be related in some manner, the event queue is emptied (step 106). Specifically, a related set of events is most likely associated with a specific action or actions that a user explicitly desires to accomplish. Accordingly, there is no necessity of offering help to a user. The event queue may be emptied by resetting the pointer defining the end of the queue thereby allowing new events to be written into the appropriate memory locations. Also, the detector algorithm returns to step 101.

[0013] If the set of events is a unrelated pattern of events, the number of events in the queue is examined (step 107). If the number of events is less than a predetermined number N, the detector algorithm returns to step 101. The predetermined number N may be selectively chosen such that the event sample length is sufficiently large to avoid interrupting the user prematurely. The predetermined number N may be adapted to an individual user in response to user queries regarding whether the user desires help. For example, the predetermined number may be changed based upon the number of times that a user responds negatively or positively to the offer for assistance. If the number of events in the queue is less than N, the detector algorithm returns to step 101.

[0014] If the number of events is equal to N, the detector algorithm queries the user to determine whether the user wants help (step 108). If the user does not desire help (step 109), the event queue is emptied (step 110) and the detector algorithm returns to step 101.

[0015] If the user does desire help, the detector algorithm initiates a search prompt (step 111). The search prompt may allow a user to enter a search topic via a keyword, keywords, or a natural language string. The search prompt may request multiple search terms and/or include boolean operators. The detector algorithm will then perform a hierarchical search based upon the user input. The detector algorithm will first search previous search results stored in a user profile (step 112). The search results from the user profile are presented to the user. The user may click on a description associated with such search results. If the user clicks on a specific result or otherwise indicates that a particular result is pertinent to the user's initial request for help (step 113), the detector algorithm proceeds to step 121.

[0016] If pertinent information is not found by searching the user profile, the detector algorithm performs a search of the help library associated with the particular application currently being operated by the user (step 114). The search results are presented to the user. If the user clicks on a specific result or otherwise indicates that a particular result is pertinent to the user's initial request for help (step 115), the detector algorithm proceeds to step 121.

[0017] If pertinent information is not found by searching the help library of the particular application currently being operated, the detector algorithm may perform a search of the help libraries of the operating system and/or other applications installed on the user system (step 116). If the user clicks on a specific result or otherwise indicates that a particular result is pertinent to the user's initial request for help (step 117), the detector algorithm proceeds to step 121.

[0018] If pertinent information is not found locally, the world wide web may be searched for the pertinent information (step 118). In particular, technical service websites and user group websites may be queried to obtain pertinent information. If the user clicks on a specific result or otherwise indicates that a particular result is pertinent to the user's initial request for help (step 119), the detector algorithm proceeds to step 121.

[0019] If no pertinent information has been identified, the detector algorithm clears the event queue (step 120) and returns to step 101. Alternatively, the search may be repeated by prompting the user for more detailed search terms in accordance with step 111.

[0020] In step 121, more detailed information associated with the selected search result or results is present to the user. In step 122, the desired information and/or the location(s) thereof are stored or otherwise designated in a user profile. The location of the information may be stored for information retained locally on the user's system. However, it may prove advantageous to cache the actual information obtained from websites for future retrieval. First, caching web content is advantageous in that it will reduce the latency associated with future retrievals. Secondly, it is possible that the location of the content may change. For example, the URL of the website may be changed before the next retrieval of the information. Caching information may ensure that valuable information is not lost.

[0021] Also, the number of times that a particular search result has been selected may be retained in the user profile. This number may be used as a weighting factor for ordering search results for presentation to the user. Additionally, the current application being operated by the user may be stored. This information may also be used as a weighting factor for ordering search results for presentation to the user.

[0022] It shall be appreciated that by utilizing a user profile, the present invention is capable of adapting to the specific needs of an individual user. The help facilities are not arbitrarily constrained. Specifically, the adaptive nature of the present invention causes categories of help information related to system functionality that is of the greatest interest to a user to be brought to the fore of search results. Thus, the user is able to quickly identify topics of interest and subsequently retrieve those topics of interest with very little effort. This aspect of the present invention is quite advantageous given the complexity of recent software programs. New software functionality need not confuse the user. Subject matter related to new software functionality may be added to a help facility without obscuring more rudimentary topics of greater interest to the user. Specifically, topics of particular interest will be retained in the user's profile thereby ensuring that additions to the help facilities do not produce confusion.

[0023] Additionally, it shall be appreciated that implementing the present invention via a modification of the operating system is quite advantageous. Specifically, many user

questions pertain to topics that possess relevance beyond a specific application. For example, a simple question directed to printing may not be completely answered by the help facilities of a specific application. Instead, useful information may be retained in the operating system help facilities. Moreover, the user question may be directed to a more pragmatic, rather than technical, concern. This type of information would more likely be set forth in a user group website, rather than in an application help facility. By preferably implementing the present invention in the operating system, the present invention may leverage each source of potentially useful information no matter which application is currently being operated by the user.

[0024] It shall further be appreciated that embodiments of the present invention are especially valuable for novice users. In particular, detecting whether a series of events presents a unrelated pattern allows help facilities to be presented to a novice user without requiring the user to perform any particular task. Thus, if it is determined that the user is lost or confused, help may be offered immediately thereby easing user frustration.

[0025] When implemented via executable instructions, various elements of the present invention are in essence the code defining the operations of such various elements. The executable instructions or code may be obtained from a readable medium (e.g., a hard drive media, optical media, EPROM, EEPROM, tape media, cartridge media, flash memory, ROM, memory stick, and/or the like) or communicated via a data signal from a communication medium (e.g., the Internet). In fact, readable media can include any medium that can store or transfer information.

[0026] FIGURE 2 depicts exemplary computer system 200 on which embodiments of the present invention may be implemented. Central processing unit (CPU) 201 is coupled to system bus 202. CPU 201 may be any general purpose CPU. Suitable processors, without limitation, include any processor from the Itanium™ family of processors, such as the McKinley processor, available from Hewlett-Packard Company, or an PA-8500 processor also available from Hewlett-Packard Company. However, the present invention is not restricted by the architecture of CPU 201 as long as CPU 201 supports the inventive operations as described herein. Additionally, it shall be appreciated that the present invention is not limited to single processor platforms. For example, embodiments of the present invention may be advantageously adapted to multi-processor systems. Computer system 200 includes random access memory (RAM) 203, which may be SRAM, DRAM, or SDRAM, as examples. Computer system 200

includes ROM 204 which may be PROM, EPROM, or EEPROM, as examples. RAM 203 and ROM 204 may hold user and system data and programs as is well known in the art.

[0027] Computer system 200 also includes input/output (I/O) adapter 205, communications adapter 211, user interface 208, and display adapter 209. I/O adapter 205 connects to storage devices 206, such as one or more of hard drive, CD drive, floppy disk drive, tape drive, to computer system 200. Communications adapter 211 is adapted to couple computer system 200 to a network 212, which may be one or more of telephone network, local (LAN) and/or wide-area (WAN) network, Ethernet network, and/or Internet network. User interface 208 couples user input devices, such as keyboard 213 and pointing device 207, to computer system 200. Display adapter 209 is driven by CPU 201 to control the display on display device 210.